Herramientas para Estimación de Campos Receptivos (RF) desde Registros de Retina in vitro utilizando arreglo Multi Electrodo (MEA).
Manual de Usuario y Referencia.

Versión 4.2.

Aland Astudillo C., aland.astudillo.c@gmail.com
Departamento de Electrónica
Universidad Técnica Federico Santa María, Valparaíso, Chile.

(14 de mayo de 2014)

Este manual sirve de guía para realizar el análisis de señales provenientes del registro de células ganglionares en retina utilizando Arreglo Multi Electrodo (Multi Electrode Array, MEA). A partir del primer análisis se obtienen los spikes por unidad (neurona o célula hipotética), y además se realiza la obtención de la respectiva señal de sincronía con el estímulo. Utilizando los spikes de las unidades válidas, la señal de sincronía y el estímulo, se pueden estimar los campos receptivos (Receptive Fields, RF). A partir de los campos receptivos estimados se pueden estudiar sus curvas temporales y perfiles espacio - temporales, y por último realizar el ajuste gaussiano de cada campo receptivo obtenido. Utilizando toda esta información es posible realizar una caracterización funcional y espacial de las células encontradas.

`aland.astudillo.c@usm.cl`.

Índice

1. DESCRIPCIÓN INICIAL

El resultado del registro MEA corresponde a un conjunto de archivos de datos de *Multi Channel System* (`*.mcd`) que contienen las señales resultantes para todos los canales de cada experimento. Es necesario tener claridad sobre el experimento realizado a través de la lectura de la bitácora (dada en formato digital o foto) generada en el experimento para poder determinar el archivo que será analizado. Para el caso de la estimación de campos receptivos se deben utilizar estímulos tipo checker board.

2. INSTRUCCIONES

2.1. OBTENER LA SEÑAL DE SINCRONÍA

Esta etapa se realiza en el programa Matlab. Desde el archivo `*.mcd` que se va a analizar se debe obtener la señal de sincronía, para ello se debe usar el script (`*.m`) de Matlab `get_synchrony_signal.m`.

Copiar el script `*.m` en la carpeta en donde se encuentra el archivo `*.mcd`. Los parámetros que se deben modificar dentro de ese script son:

- pathname
- filename
- filename2 (realizar el cambio respectivo para usar sólo una variable en vez de dos)

En `pathname` debe ir la ruta completa del lugar donde se encuentra el archivo `*.mcd`, por ejemplo
`pathname = 'C:\Users\ALIEN3\Desktop\resultados_26-12\';`

En `filename` debe ir el nombre del archivo `*.mcd` (con la extensión incluida) del cual se desea extraer la señal de sincronía, por ejemplo
`filename = 'datos0005.mcd';`

En `filename2` va el nombre del archivo sin la extensión (parámetro eliminado para próxima versión), por ejemplo
`filename2 = 'datos0005';`

Previo a la ejecución del script `get_synchrony_signal.m`, se debe añadir al directorio de *Matlab* la carpeta **"ns"**, la cual contiene las funciones de *Neuroshare* para permitir leer archivos del tipo `*.mcd` desde *Matlab* (o *Python*). También se debe asegurar de tener en esa carpeta el archivo `*.dll` específico (librería de *Neuroshare*) para el sistema operativo utilizado (caso de Sistema Operativo *Windows*) o el archivo `*.so` (caso de *Unix*). Existe una versión del `get_synchrony_signal` utilizado en *Python* que puede realizar esta misma función para obtener la señal de sincronización (normalmente el canal 252 del archivo `*.mcd`), pero no está documentada en esta guía.

Luego de añadir al path la carpeta `ns` y modificar los campos mencionados, se puede ejecutar el script `get_synchrony_signal.m` presionando F5 o con el botón Run. La salida del programa corresponde a un archivo `*.mat` con la señal de sincronía y una imagen de la señal en formato `*.pdf`. El nombre del archivo de salida del script `get_synchrony_signal.m` quedará como
`syn_signal1_xxxxx.mat`,

donde la última porción del nombre `xxxxx` corresponde al mismo nombre definido como el parámetro `filename2`.

2.2. ANALIZAR LA SEÑAL DE SINCRONÍA

Esta etapa se realiza en *Matlab*. La señal obtenida en la etapa anterior es analizada en búsqueda de las posiciones exactas de cada frame en el tiempo. Se utiliza el archivo `*.mat` generado como `syn_signal1_xxxxx.mat`, donde `xxxxx` corresponde al nombre del archivo `*.mcd` analizado, sin la extensión. El script para realizar lo descrito se llama `sync_signal_analizer3.m`.

Se debe copiar este script en la carpeta en donde se encuentra el archivo `*.mcd` y donde se encuentra el archivo `*.mat` generado anteriormente. Los parámetros que se deben definir y cambiar en este script son

`datosname`

En `datosname` debe ir el nombre del archivo `*.mcd` (sin la extensión), por ejemplo

```
datosname = 'datos0005';
```

Ejecutar el script presionando F5 o el botón Run. Cuando se ejecute el script se mostrarán en pantalla los valores de promedio y desviación estándar de las distancias entre frames, en puntos. En este caso para 20kHz, e imágenes a 60Hz (frame rate), la distancia promedio entre frames es de 334 puntos y una desviación estándar de 2 puntos (aproximadamente). Si esto no se cumple, entonces quiere decir que existe un error en la detección de los tiempos de los frames en la señal de sincronización, y el análisis no se podrá hacer. Para corregir esto se debe cambiar el parámetro de búsqueda dentro de los ciclos de análisis de posiciones.

La salida del script corresponde a un archivo llamado

`inicio_fin_frame_xxxxx.txt`,

en donde `xxxxx` corresponde al nombre del archivo `*.mcd` analizado, determinado por el parámetro `datosname`. Este archivo de texto contiene el tiempo de inicio y fin de cada frame, en puntos (en dos columnas).

2.3. ANALIZAR DATOS DE REGISTRO: Ordenamiento de Espigas (SPIKE SORTING)

Esta etapa se realiza utilizando *Offline Sorter* (v3.3) de *Plexon*, sólo disponible en sistema operativo *Windows*.

La entrada corresponde al archivo `*.mcd` completo. Se recomienda que una vez cargado el archivo `*.mcd`, este sea guardado y transformado a formato *Plexon* `*.plx`, lo cual permite trabajar de forma más rápida en la lectura de canales y la realización de todos los análisis de spike sorting realizados en *Offline Sorter*.

El análisis consiste en 4 etapas: filtrado de la señal, detección de spikes, extracción de características (creación de sub espacio) y selección, y finalmente el clustering (agrupación en el sub espacio de caracteristicas creado). La primera etapa corresponde al filtrado de la señal (filtro pasa alto 100Hz Butterworth) en cada canal del multi electrodo (252 electrodos). La detección de los spikes (umbral inferior para detección de los spikes en su parte negativa) se realiza utilizando como medida umbral un factor que depende de la desviación estándar del ruido estimado o simplemente de un valor extraído desde la señal. Los spikes encontrados deben ser alineados o centrados y luego se realiza una extracción de características, utilizando Análisis de Componentes Principales (Principal Component Analysis, PCA). Finalmente se realiza una etapa de clustering (manual o automática, dependiendo del método seleccionado, entre los cuales existe K-Means, Valley Seek, T Distribution - Expectation Maximization, entre otros). El resultado corresponde a varios grupos o clústers, donde cada clúster equivale a una unidad o potencial célula a identificar.

El archivo resultante contendrá las señales continuas y además, los resultados de las unidades encontradas y sus respectivos tiempos de spikes. Existen varias opciones para realizar el guardado del archivo. Uno puede guardar el archivo completo o existe la opción de sólo guardar los spikes encontrados y sus respectivos tiempos. Una vez terminado el análisis, se guarda un archivo `*.plx`, que contiene SÓLO los spikes y tiempos de spikes de las unidades celulares encontradas. El archivo `*.plx` debe ser nombrado como

`xxxx_ts_fecha.plx`,

en donde `xxxx` corresponde al nombre del archivo `*.mcd` original, y fecha corresponde a la fecha del análisis, en formato dd-mm-aaaa.

Nota: Existe una versión de Spike Sorting desarrollado localmente en *Python*, denominado SSpy, pero esa versión no está documentada en esta guía. La implementación es funcional pero necesita una etapa de interfaz más potente y amistosa para poder realizar mejor las tareas necesarias de selección de grupos, etc. El código está disponible para su mejoramiento y utilización.

Nota: Existe una versión de Spike Sorting desarrollado localmente en *Matlab*, denominado SSmat (símil de SSpy), pero no está documentado en esta guía. La implementación es funcional y escalable, pero falta una etapa de interfaz más amigable para realizar las selecciones de grupos de manera más amena.

2.4. GUARDAR TIEMPOS DE SPIKES

Utilizando el programa *NeuroExplorer* (en *Windows*), abrir el archivo `*.plx` guardado con los spikes y tiempos de spikes de las unidades obtenidas en la etapa anterior. Ir a la pestaña de **timestamps** y copiar toda la tabla de datos que aparece. Crear un archivo de texto nuevo (Block de notas, Notepad++) pegar los datos y guardar con extensión `*.txt`. Guardar idealmente con el mismo nombre o similar a `xxxx_timestamps_fecha.txt`, donde `xxxx` y `fecha` dependen del archivo de origen de este resultado (`datos00xx` y `fecha`).

Abrir el script `readwrite_timestamps_txt.py` con un editor de texto. Editar los parámetros `tsfolder` y `filetimestamps`. El parámetro `tsfolder` corresponde a la carpeta en donde se guardarán los tiempos de spike de cada unidad. Cada unidad tendrá sus tiempos de spikes en archivos de texto `*.txt` aislados, por ejemplo

```
tsfolder = 'TS_datos0003'
```

El segundo parámetro `filetimestamps` corresponde al nombre del archivo de texto en donde se guardaron los tiempos de spikes desde el archivo `*.plx` en el paso anterior

```
filetimestamps = 'xxxx_timestamps_fecha.plx'
```

donde `xxxx` y `fecha` dependen del archivo real revisado. Guardar los cambios y ejecutar desde command Windows (o desde terminal)

```
>> python readwrite_timestamps_txt.py
```

Se creará la carpeta respectiva (según el nombre definido por el parámetro tsfolder) y dentro se crearán los archivos `*.txt` para cada unidad o célula hipotética.

2.5. SPIKE TRIGGERED AVERAGE (STA)

El Spike Triggered Average (STA) corresponde a la estimación lineal de los campos receptivos celulares utilizando los tiempos de disparo de las unidades de células y el estímulo que gatillo cada disparo. El STA más básico logra una representación lineal del campo receptivo, y corresponde a una descripción estadística promedio de todos los posibles estímulos que pueden gatillar disparo para una célula específica.

Para realizar el STA se pueden utilizar dos opciones. La primera opción es usar un código para realizar el STA para sólo una unidad específica. La segunda forma es utilizar un segundo código para realizar el STA a un conjunto de varias unidades analizando de manera consecutiva.

2.5.1. STA INDIVIDUAL: stapy

Para utilizar la primera forma, mediante la cual se analiza sólo una unidad seleccionada arbitrariamente, se utiliza el script `sta_v4_sspy_xxxx.py`, en donde xxxx debe ser reemplazado por el nombre del archivo `*.mcd` analizado, sin la extensión `*.mcd`. Copiar el script `sta_v4_sspy_xxxx`.py en la carpeta en donde se encuentre la carpeta `tsfolder`, y el archivo `inicio_fin_frame_xxxx.txt`. Además, es necesario conocer el lugar en donde se encuentra la carpeta contenedora de las imágenes (estímulos, imágenes `*.png` o `*.jpg` guardadas según índice). En este ejemplo la carpeta contenedora de las imágenes es denominada `checkImages`.

La primera vez que se realiza el STA, se pueden hacer un par de pruebas de lectura de archivos (opcionalmente). Sin embargo, la primera vez se deben obtener la imagen promedio (`mean_image.mat`) y generar un archivo que contiene los nombres de las direcciones de todas las imágenes (`image_filenames.mat` y `image_filenames.txt`).

Abrir el archivo script y editar los siguientes parámetros

- `stafolder` : corresponde a la carpeta que contendrá los resultados del STA, por ejemplo

  ```
  stafolder = 'STA_datos0003'
  ```

- `imageruta` : corresponde a la ruta o path en donde se encuentra la carpeta que contiene las imágenes o estímulos, por ejemplo

  ```
  imageruta = 'D:/'
  ```

- `imagefolder` : corresponde al nombre de la carpeta contenedora de imágenes, por ejemplo

  ```
  imagefolder = 'checkImages'
  ```

- `imagefiltro` : corresponde al tipo o extensión de las imágenes que serán cargadas, por ejemplo

  ```
  imagefiltro = '*.png'
  ```

- `timefolder` : corresponde al nombre de la carpeta que contiene las unidades y sus respectivos timestamps, por ejemplo

```
timefolder = 'TS_datos0003/'
```

- `samplingRate` : corresponde a la tasa de muestreo en Hz de los datos adquiridos originalmente, por ejemplo

```
samplingRate = 20000
```

- `numberframes` : corresponde al número de frames antes del spike que serán utilizados para obtener el STA, por ejemplo

```
numberframes = 13
```

- `numberframespost` : corresponde al número de frames después del spike que serán utilizados para calcular el STA, por ejemplo

```
numberframespost = 5
```

- `synchronyfile` : corresponde al nombre del archivo de texto que contiene los índices de tiempos de los frames, y que es el resultado del análisis de la señal de sincronía, visto en una de las etapas previas de este manual, por ejemplo

```
synchronyfile = 'inicio_fin_frame_datos0003.txt'
```

- `sizex` : corresponde al tamaño en el eje x de la imagen a cargar, en pixeles, por ejemplo

```
sizex = 380
```

- `sizey` : corresponde al tamaño en el eje y de la imagen a cargar, en pixeles, por ejemplo

```
sizey = 380
```

- `dolog` : corresponde a la opción de guardar los resultados del STA escalados usando logaritmo (no utilizado, próxima versión estará obsoleto) , por ejemplo

```
dolog = 0
```

Una vez editados estos parámetros, guardar los cambios y ejecutar desde command Windows (o terminal). El script necesita 5 parámetros de entrada:

- `getimagenames` : se utiliza la primera vez como parámetro activo para generar el archivo con la lista de los nombres y path de las imágenes.

- `openimagesandwrite` : corresponde a un test para leer y escribir imágenes usando el paquete respectivo (opcional).

- `calculatemeanrf` : se utiliza la primera vez como parámetro activo para calcular el estímulo promedio desde todas las imágenes.

- `tipoalgoritmo` : corresponde al tipo de algoritmo a utilizar (cargar conjunto de imágenes completo o realizar una carga secuencial). El tipo de algoritmo utilizado actualmente corresponde al 2, ya que el primero está obsoleto momentáneamente.

- `timestampName` : corresponde al nombre de la unidad a analizar, por ejemplo si la unidad a analizar corresponde a Ç12a", se ingresa esa palabra.

Para ejecutar el código por primera vez para analizar la unidad C12a se escribe en command windows o en terminal

```
>> python sta_v4_sspy_xxxx.py 1 1 1 2 C12a
```

Luego de que se cumplan todas las etapas, se creará una carpeta denominada C12a dentro de la carpeta de STA (según el nombre definido como parámetro) que contendrá un número de frames y el promedio de esos frames. Además, la matriz STA y otros parámetros estarán guardados como archivos `*.mat`. Cuando la unidad efectivamente corresponde a una célula, la dinámica espacio temporal del campo receptivo podrá ser visualizada en los diferentes frames resultantes.

Para ejecutar el código, luego para cualquier otra unidad, se puede escribir

```
>> python sta_v4_sspy_xxxx.py 0 0 0 2 G8a
```

Los parámetros iniciales ya no son necesarios, ya que tanto el archivo de imagen promedio (`mean_image.mat`) y el archivo con la lista de imágenes (`image_filenames.txt`) ya fueron creados. Cada vez que se realice el análisis en un computador se debe crear una sola vez estos archivos. Si para varios análisis de STA de diferentes fechas las imágenes son las mismas utilizadas (es decir que la semilla para generar el checkerboard es la misma) es posible saltarse el paso de obtener la imagen promedio y la ruta de todas las imágenes, y saltar al paso de realizar el STA sin ningún problema. Sin embargo, se debe considerar siempre que para distintas fechas de análisis (un archivo de uno u otro experimento) se tendrán señales de sincronía distintas. Por lo tanto, el análisis de la señal de sincronía no es opcional, y se debe realizar siempre para cada archivo analizado.

Los archivos de salida que genera el script STA estarán contenidos en la carpeta STA definida por parámetro. Dentro de la carpeta estarán las carpetas de unidades, cuyos nombres son los respectivos a cada unidad analizada. Dentro de cada carpeta para cada unidad se encontrará la colección de frames en formato `*.png` correspondientes a la estimación del campo receptivo en una representación espacio temporal (ver figura 1), el STA promedio, y archivos `*.mat` que contienen la matriz STA escalada para visualización y la matriz STA sin escalar; otro archivo `*.mat` conteniendo cada timestamp e índice de frame respectivo (para verificación). Los archivos se denominan `stavisual_lin_array_xxxx.mat` y `spikeframe_matrixxxxx.mat`, respectivamente, donde `xxxx` corresponde al nombre de la unidad analizada.

Nota: El tiempo de cálculo de un STA dependerá, entre otros, de la cantidad de spikes que tiene la unidad analizada. En promedio, un análisis de este tipo para una unidad que tiene 1000 spikes, puede tomar entre 10 a 15 minutos (Intel I7 2.GHz 8GB RAM, Windows7).

Nota: Existe una versión del STA actual que mejora la eficiencia del código. En lugar de leer las imágenes desde un directorio, carga un archivo `*.mat` con el estímulo comprimido a su tamaño original.

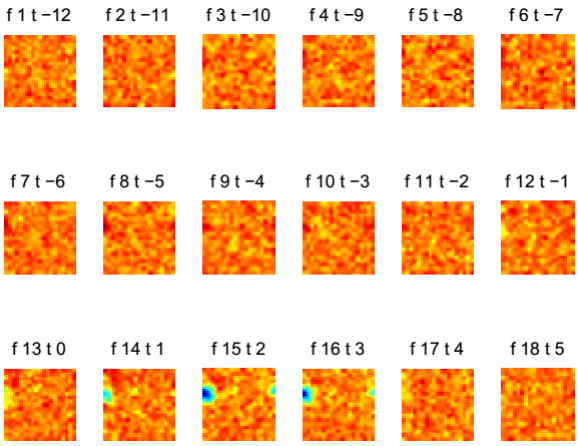

Figura 1: Colección de frames del STA obtenido para una unidad en particular. En este caso es posible decir que efectivamente aparece un campo receptivo.

Esto permite mejorar enormemente la eficiencia, disminuyendo el tiempo requerido para cada análisis. Esta versión estará disponible pronto.

Nota: La versión actual está siendo modificada para hacer más ameno el ingreso de variables y la definición de parámetros opcionales.

2.5.2. STA GRUPAL: stapy chain

Para ejecutar el análisis STA de un bloque o conjunto de unidades se debe usar el script `sta_chain2_xxxx.py` (se debe tener en la misma carpeta el archivo `sta_functions2.py`), donde `xxxx` es reemplazado por el nombre del archivo `*.mcd` analizado. Copiar y pegar el script `sta_chain2_xxxx.py` en la misma carpeta en donde se encuentra la carpeta que contiene los resultados, en este ejemplo la carpeta se llama `TS_datos0003`, por lo tanto los tiempos de disparo provienen del análisis del archivo denominado `datos0003.mcd`, y por lo tanto el script queda como `sta_chain2_datos0003.py`. Editar el script y modificar los siguientes parámetros

- `archivosruta` : corresponde a la ruta completa de donde se encuentra la carpeta que contiene los timestamps aislados en archivos `*.txt`, por ejemplo

  ```
  archivosruta = 'D:/Experimentos_CINV/datos_20-11-2013_bga50um_2/'
  ```

- `archivosfolder` : corresponde al nombre de la carpeta que contiene los timestamps de las unidades a analizar, por ejemplo

  ```
  archivosfolder = 'TS_datos0003/'
  ```

- `archivofiltro` : corresponde a la extensión de los archivos que contienen los tiempos de spike (timestamps) de las unidades, y que funciona como filtro para cargar los archivos, en este caso se utiliza por defecto

  ```
  archivofiltro = '*.txt'
  ```

Los parámetros `getimagenames`, `openimagesandwrite`, `calculatemeanrf` y `tipoalgoritmo` corresponden a los mismos definidos en el caso anterior y si no es primera vez que se ejecuta el código se pueden dejar como

- `getimagenames = 0`
- `openimagesandwrite = 0`
- `calculatemeanrf = 0`
- `tipoalgoritmo = 2`

Los siguientes parámetros son similares a los definidos en el caso anterior, y se pueden dejar, por ejemplo

- `stafolder = 'STA_datos0003_2'`
- `imageruta = 'D:/'`
- `imagefolder = 'checkImages'`
- `imagefiltro = '*.png'`
- `samplingRate = 20000`
- `numberframes = 18`
- `numberframespost = 2`
- `synchronyfile = 'inicio_fin_frame_datos0003.txt'`
- `sizex = 380`
- `sizey = 380`
- `dolog = 0`

Los últimos parámetros que se editan son inicio y final, con los cuales se define el intervalo de índices de las unidades que serán analizadas. Para saber el índice de una unidad, se debe tener en paralelo una tabla Excel en donde se ordenan las unidades con sus nombres e índices (recomendado). Estos parámetros pueden quedar, por ejemplo

```
inicio = 10 -1
final  = inicio - - 5
```

lo cual significa que se analizará desde la unidad 10 hasta la unidad 10+5, es decir 5 unidades, cuyos nombres se pueden ver en la tabla Excel según su índice. Al guardar los cambios y ejecutar, se obtendrá la creación de las carpetas respectivas y el análisis de cada unidad.

Nota: El script `sta_chain2_xxxx.py` utiliza al script `sta_functions2.py`. Este script se ha modificado para permitir guardar la matriz STA real (cuyas amplitudes no han sido reescaladas, y es la que sirve) y la matriz STA re escalada para visualización. Esto no sucede así en el caso del script que analiza las unidades de manera individual, pero para la siguiente versión si estará incluido para ambos casos.

Nota: Existe un formato para la generación de la tabla Excel que agrupa y ordena los resultados que se van obteniendo en cada uno de los análisis, y que además contiene el resultado del análisis por inspección de los campos receptivos obtenidos. Si bien este análisis es subjetivo, puede apoyarse por otros tipos de análisis para determinar si efectivamente el resultado del proceso de STA resulta en un campo receptivo real o no, y sirve como apoyo para los próximos pasos en el análisis.

2.5.3. STA Rápido: stapy FAST

Usando una versión reducida del conjunto de estímulo, y en lugar de las imagenes en tamaño normal de presentación (380x380px), usar un archivo que contiene todos los frames en una resolución menor, sin perder las características que se pueden observar (es decir que la forma de cada bloque del checkerboard permanezca intacta, es posible obtener una versión del STA que demora mucho menos en el cálculo.

Para ejecutar el análisis STA de un bloque o conjunto de unidades se debe usar el script `STA_FAST.py`. Copiar y pegar el script `STA_FAST.py` en la misma carpeta en donde se encuentra la carpeta que contiene los resultados del ordenamiento de spikes, en este ejemplo la carpeta se llama `TS_datos0003`, por lo tanto los tiempos de disparo provienen del análisis del archivo denominado `datos0003.mcd`. Los parámetros de entrada son los siguientes:

- `archivosruta` : corresponde a la ruta completa de donde se encuentra la carpeta que contiene los time stamps aislados en archivos `*.txt`, por ejemplo

  ```
  archivosruta = 'D:/Experimentos_CINV/datos_20-11-2013_bga50um_2/'
  ```

- `archivosfolder` : corresponde al nombre de la carpeta que contiene los time stamps de las unidades a analizar, por ejemplo

  ```
  archivosfolder = 'TS_datos0003/'
  ```

- `archivofiltro` : corresponde a la extensión de los archivos que contienen los tiempos de spike (timestamps) de las unidades, y que funciona como filtro para cargar los archivos, en este caso se utiliza por defecto

  ```
  archivofiltro = '*.txt'
  ```

- `getimagenames` : corresponde a la realización de una carga de imágenes como prueba del paquete específico (en la próxima versión estará obsoleto). En este caso se deja como

  ```
  getimagenames = 0
  ```

- `openimagesandwrite` : corresponde a una prueba del módulo (en la próxima versión estará obsoleto). En este caso se deja como

  ```
  openimagesandwrite = 0
  ```

- `calculatemeanrf` : corresponde al cálculo del estímulo medio, sólo para el caso en que se cargan las imágenes desde una carpeta contenedora (en la próxima versión estará obsoleto). Para este caso se deja como

  ```
  calculatemeanrf = 0
  ```

- `tipoalgoritmo` : corresponde al tipo de algoritmo a utilizar para calcular el STA. En este caso se utiliza el algoritmo tipo 4, para cargar un conjunto estímulo en un archivo `*.mat`

  ```
  tipoalgoritmo = 4
  ```

- `stafolder` : corresponde a la carpeta que contendrá los resultados del STA, por ejemplo

  ```
  stafolder = 'STA_datos0003'
  ```

- `imageruta` : corresponde a la ruta o path en donde se encuentra la carpeta que contiene las imágenes o estímulos, por ejemplo

  ```
  imageruta = 'D:/'
  ```

- `imagefolder` : corresponde al nombre de la carpeta contenedora de imágenes, por ejemplo

  ```
  imagefolder = 'checkImages'
  ```

- `imagefiltro` : corresponde al tipo o extensión de las imágenes que serán cargadas, por ejemplo

  ```
  imagefiltro = '*.png'
  ```

- `timefolder` : corresponde al nombre de la carpeta que contiene las unidades y sus respectivos timestamps, por ejemplo

  ```
  timefolder = 'TS_datos0003/'
  ```

- `samplingRate` : corresponde a la tasa de muestreo en Hz de los datos adquiridos originalmente, por ejemplo

  ```
  samplingRate = 20000
  ```

- `numberframes` : corresponde al número de frames antes del spike que serán utilizados para obtener el STA, por ejemplo

  ```
  numberframes = 13
  ```

- `numberframespost` : corresponde al número de frames después del spike que serán utilizados para calcular el STA, por ejemplo

  ```
  numberframespost = 5
  ```

- `synchronyfile` : corresponde al nombre del archivo de texto que contiene los índices de tiempos de los frames, y que es el resultado del análisis de la señal de sincronía, visto en una de las etapas previas de este manual, por ejemplo

  ```
  synchronyfile = 'inicio_fin_frame_datos0003.txt'
  ```

- `sizex` : corresponde al tamaño en el eje x de la imagen a cargar, en pixeles, por ejemplo

  ```
  sizex = 380
  ```

- `sizey` : corresponde al tamaño en el eje y de la imagen a cargar, en pixeles, por ejemplo

  ```
  sizey = 380
  ```

- `dolog` : corresponde a la opción de guardar los resultados del STA escalados usando logaritmo (no utilizado, en la próxima versión estará obsoleto) , definir como

  ```
  dolog = 0
  ```

- `stim_mini` : corresponde al nombre del archivo `*.mat` que contiene el conjunto de estímulos en una versión reducida. En este caso dejar como

  ```
  stim_mini = stim_mini.mat
  ```

Los últimos parámetros que se editan son inicio y final, con los cuales se define el intervalo de índices de las unidades que serán analizadas. Para saber el índice de una unidad, se recomienda contar con un archivo Excel en donde se pueda revisar en paralelo las unidades con sus nombres y números. Los parámetros de inicio y fin pueden quedar, por ejemplo

```
inicio = 10 -1
final  = inicio - - 5
```

lo cual significa que se analizará desde la unidad 10 hasta la unidad 10+5, es decir 5 unidades, cuyos nombres se podrían ver en la tabla Excel según su índice. Al guardar los cambios y ejecutar, se obtendrá la creación de las carpetas respectivas y el análisis de cada unidad.

Nota: Este script contiene las modificaciones respectivas para cargar el conjunto de estímulos desde un archivo mat. Además, se ha modificado para permitir guardar la matriz STA real (cuyas amplitudes no han sido re escaladas para el análisis) y la matriz STA re escalada para visualización.

Nota: Existe un formato para la generación de la tabla Excel que agrupa y ordena los resultados que se van obteniendo en cada uno de los análisis, y que además contiene el resultado del análisis por inspección de los campos receptivos obtenidos. Si bien este análisis es subjetivo, puede apoyarse por otros tipos de análisis para determinar si efectivamente el resultado del proceso de STA resulta en un campo receptivo real o no, y sirve como apoyo para los próximos pasos en el análisis.

Nota: Es posible que esta versión reemplace finalmente a las dos versiones anteriores debido al menor costo en cálculos y tiempo que requiere. Por esta razón, en la próxima versión del manual ya no se incluirán stapy ni stapy chain.

2.6. VALORIZACIÓN DE LOS CAMPOS RECEPTIVOS ESTIMADOS

No siempre ocurre que se obtienen campos receptivos verdaderos cuando se analiza un tren de spikes y el estímulo respectivo. Por lo tanto, es necesario revisar cada resultado para cada tren de spikes e inspeccionar y declarar si efectivamente cada archivo resultante corresponde o no a un campo receptivo real y perceptible. Como esta tarea de inspección de campos receptivos es subjetiva, es necesario contar con una valorización objetiva que simplifique la tarea de inspección, y facilite esta tarea. Por este motivo se tiene el código `sta_inspector.m`, el cual arroja un valor porcentual de calidad de STA realizado en función de los valores peak y variaciones de cada frame del STA.

Los parámetros de este script son
`sta_folder` : corresponde al nombre de la carpeta que contiene los resultados del STA, por ejemplo

```
sta_folder= 'STA_datos0003_2/';
```

`ts_folder` : corresponde al nombre de la carpeta que contiene los archivos de time stamps, por ejemplo

```
ts_folder= 'TS_datos003_2/'];
```

El script analizará todos los archivos contenidos de matriz de campo receptivo estimado resultantes del STA y arrojará un vector en donde valoriza la calidad de cada STA. Aquellos STA con valores bajo cierto umbral, no corresponderán a un campo receptivo perceptible, por lo tanto sólo se inspeccionan aquellos STA que si cumplen con el umbral.

2.7. REALIZAR AJUSTE GAUSSIANO DE LOS CAMPOS RECEPTIVOS

Para cada STA de las unidades analizadas en los pasos anteriores, es posible realizar el ajuste gaussiano y análisis de su dinámica temporal, con el objetivo de caracterizar tanto temporal como espacialmente (tamaño) cada unidad o célula identificada a través de su campo receptivo. Cada campo receptivo que cumple los requerimientos según inspección visual o por análisis. El ajuste gaussiano se realiza sólo para uno de los frames del STA (o para todos). Se utiliza una gaussiana de dos dimensiones para ajustar a la forma vista en aquel frame principal que presente el valor extremo máximo (intensidad en pixel) o mínimo de todos los frames del STA.

Para estudiar la curva temporal, se busca la posición en pixel del punto extremo máximo (o mínimo) en intensidad y luego se revisa su variación para todos los frames en el mismo punto. Esto arroja una curva que nos dice el comportamiento temporal principal del campo receptivo, con lo cual es posible caracterizar temporalmente la neurona estudiada, lo cual coincide con las descripciones neurobiológicas de neuronas sensoriales. En el caso de las células ganglionares, de manera general es posible clasificarlas como ON/OFF dependiendo de la forma de la curva temporal.

Para llevar a cabo este análisis se utiliza el script en Matlab `gauss2dfitSTA.m`, cuyos parámetros de configuración son

`nombre_cell_grupo` : corresponde al nombre de la unidad a analizar para el ajuste gaussiano, por ejemplo

```
nombre_cell_grupo = 'A2a';
```

`carpeta` : corresponde al nombre de la carpeta que contiene los resultados del STA, y se debe reemplazar sólo la primera parte del nombre, por ejemplo

```
carpeta = ['STA_datos0003_2/',nombre_cell_grupo,'_lineal/'];
```

Una vez editados los parámetros ejecutar el script presionando F5 o el botón Run. Este script utiliza internamente la función `fmgaussfit()` (REF) para realizar el ajuste de una gaussiana de dos dimensiones. La salida de esa función resulta en los parámetros de ajuste de la gaussiana, la gaussiana de dos dimensiones, datos para graficar, y el error del ajuste. Además, se utiliza el comando `ellipse()` (REF) para crear una elipse con los datos de desviación estándar en las direcciones de los semiejes y el ángulo de giro (ver figura 2), resultantes del ajuste gaussiano.

Las salidas del script corresponden a 4 imágenes (en formato `*.pdf`) y un archivo `*.mat` con parámetros. La primera imagen corresponde a una vista de todos los frames del STA sin ninguna modificación. La segunda imagen corresponde al perfil 3D del frame de STA original y ajustado, y los perfiles unidimensionales laterales (x, y) del ajuste (ver figura 2). La tercera imagen corresponde a los perfiles 2D del frame de STA original y ajustado, y las elipses respectivas (un factor de la desviación estándar en cada eje y el ángulo de la gaussiana). La cuarta imagen corresponde a la curva temporal obtenida (ver figura 3). La curva temporal se construye usando los valores de intensidad de un pixel en particular, en todos los frames del STA. La posición del pixel se elige en base a aquel pixel con el valor extremo máximo (o mínimo) de todos los frames del STA. El archivo `*.mat` guardado se denomina `fit_var.mat`, y guarda todos los parámetros y datos del frame de STA ajustado.

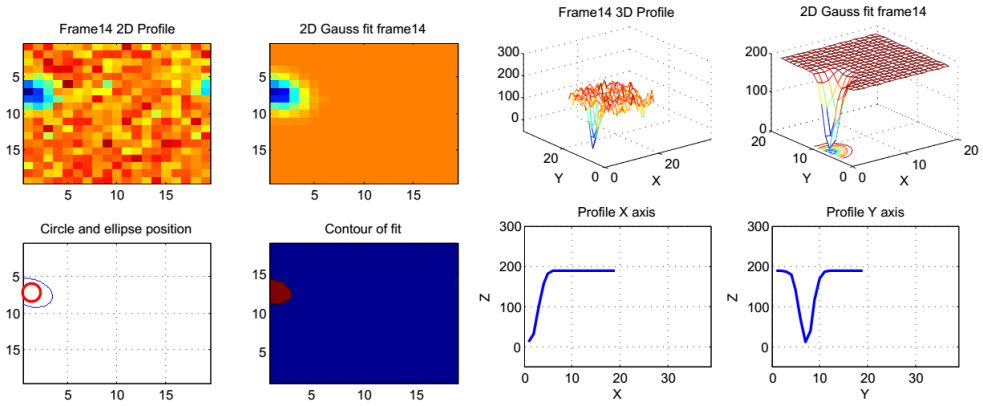

Figura 2: Ejemplo de ajuste de gaussian de dos dimensiones a los datos de un frame del STA correspondientes a una unidad cuyo campo receptivo es válido. Izquierda, perfil de dos dimensiones del campo receptivo real (un frame) y su ajuste, mostrando además las elipse respectiva a los parámetros resultantes del ajuste. Derecha, perfiles 3D del frame real y del ajuste gaussiano.

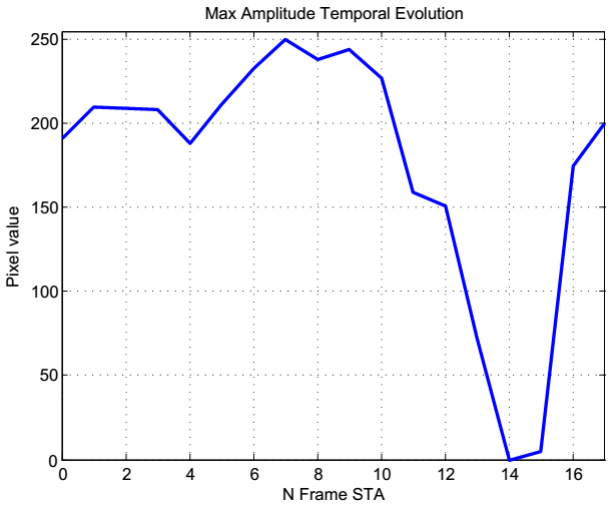

Figura 3: Evolución temporal del pixel principal del STA. La posición del pixel principal es elegida de manera que contiene el valor extremo máximo (o mínimo) de todos los pixeles en la matrix STA.

Referencias

[1] A. M. Litke, N. Bezayiff, E. J. Chichilnisky, W. Cunningham, W. Dabrowski, A. A. Grillo, M. Grivich, P. Grybos, P. Hottowy, S. Kachiguine, R. S. Kalmar, K. Mathieson, D. Petrusca, M. Rahman, and A. Sher, *What Does the Eye Tell the Brain?: Development of a System for the Large-Scale Recording of Retinal Output Activity.* IEEE Transactions on Nuclear Science, 51 (4) (2004).

[2] G.D. Field and E.J. Chichilnisky, *Information Processing in the Primate Retina: Circuitry and Coding.* The Annual Review of Neuroscience. (30) (2007).

[3] Greg D. Field, Alexander Sher, Jeffrey L. Gauthier, Martin Greschner, Jonathon Shlens, Alan M. Litke, and E. J. Chichilnisky, *Spatial Properties and Functional Organization of Small Bistratified Ganglion Cells in Primate Retina.* The Journal of Neuroscience 27 (48) (2007).